EUROPEAN PATENT OFFICE

Patent Abstracts of Japan

PUBLICATION NUMBER : 04354068
PUBLICATION DATE : 08-12-92

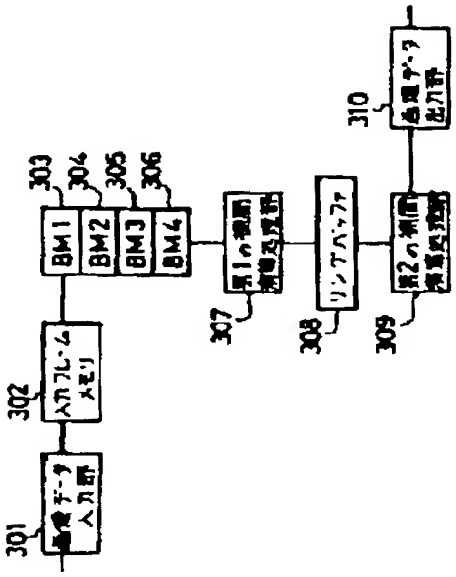
APPLICATION DATE : 31-05-91
APPLICATION NUMBER : 03129445

APPLICANT : KONICA CORP;

INVENTOR : HANDA HIDEYUKI;

INT.CL. : G06F 15/66 B41J 2/485 G06F 15/353 H04N 1/393

TITLE : METHOD AND DEVICE FOR INTERPOLATING PICTURE DATA



ABSTRACT : PURPOSE: To reduce a picture memory to be required, and in addition, to realize high-speed processing at the time when two-dimensional picture data is interpolated by executing one-dimensional interpolation twice in a first and a second directions.

CONSTITUTION: The picture data inputted from a picture data input part 301 is interpolation-arithmetic-processed in the first direction (vertical direction) by a first interpolation arithmetic processing part 307. From the point of time when this interpolation arithmetic processing proceeds to a state that the interpolation arithmetic processing in the second direction (horizontal direction) is possible, the picture data is interpolation-arithmetic-processed in parallel in the second direction by a second interpolation arithmetic processing part 309, and the picture data after the interpolation is outputted from a picture output part 310.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平4－354068

(43)公開日　平成４年(1992)12月８日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/66	３５５　Ｃ	8420－5L		
Ｂ４１Ｊ　2/485				
Ｇ０６Ｆ　15/353		6798－5L		
Ｈ０４Ｎ　1/393		8839－5C		
		8804－2C	Ｂ４１Ｊ　3／12　Ｇ	

審査請求　未請求　請求項の数６(全 12 頁)

(21)出願番号　特願平3－129445

(22)出願日　平成３年(1991)５月31日

(71)出願人　000001270
コニカ株式会社
東京都新宿区西新宿１丁目26番２号

(72)発明者　石井　満
東京都日野市さくら町１番地　コニカ株式会社内

(72)発明者　半田　英幸
東京都日野市さくら町１番地　コニカ株式会社内

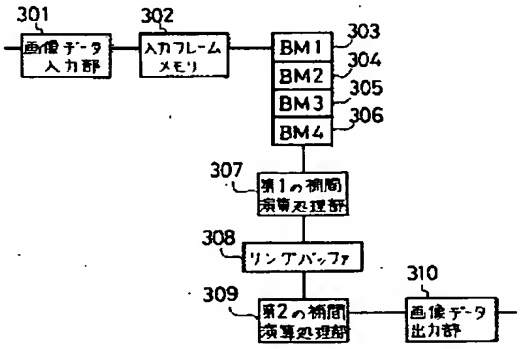
(74)代理人　弁理士　笹島　富二雄

(54)【発明の名称】　画像データ補間方法及び装置

(57)【要約】

【目的】　２次元画像データを１次元の補間を第１及び第２の方向に２度行うことによって補間するに際し、必要となる画像メモリを少なくし、かつ、高速の処理を実現させる。

【構成】画像データ入力部301 より入力された画像データを第１の補間演算処理部307により第１の方向（縦方向）に補間演算処理し、この補間演算処理が第２の方向（横方向）の補間演算処理が可能となるところまで進んだ時点から、並行して、第２の補間演算処理部309 により第２の方向に補間演算処理して、画像データ出力部310 から補間後の画像データを出力する。



1

【特許請求の範囲】

【請求項1】２次元画像データを１次元の補間を第１及び第２の方向に２度行うことによって補間するに際し、入力された画像データを第１の方向に補間演算処理し、この第１の方向の補間演算処理が第２の方向の補間演算処理が可能となるところまで進んだ時点から並行して第２の方向に補間演算処理することを特徴とする画像データ補間方法。

【請求項2】前記第２の方向が画像データ出力方向と一致していることを特徴とする請求項１記載の画像データ補間方法。

【請求項3】前記第１の方向が画像データ出力方向と一致していることを特徴とする請求項１記載の画像データ補間方法。

【請求項4】２次元画像データを１次元の補間を第１及び第２の方向に２度行うことによって補間する画像データ補間装置において、画像データ入力部と、入力された画像データを第１の方向に補間演算処理する第１の補間演算処理部と、第１の方向の補間演算処理が第２の方向の補間演算処理が可能となるところまで進んだ時点から並行して第２の方向に補間演算処理する第２の補間演算処理部と、第２の方向の補間演算処理を終えた画像データを出力する画像データ出力部と、を有することを特徴とする画像データ補間装置。

【請求項5】前記第２の方向が画像データ出力方向と一致していることを特徴とする請求項４記載の画像データ補間装置。

【請求項6】前記第１の方向が画像データ出力方向と一致していることを特徴とする請求項４記載の画像データ補間装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、画像処理装置における画像データ補間方法及び装置に関するもので、特に階調を有するデジタル化された医用画像において好適に利用しうる。

【０００２】

【従来の技術】画像をプリントアウトする場合など、粗く取られた画像データに補間を施して、より細かなデータに変換し、高精細な画像を得ることが知られている。一般的な補間の方法としては、図10又は図11に示すようなものがある。図中、●は元々値の与えられている点、○はこれから値を求めようとしている点である。

【０００３】図10は近くの点を代用する方法である。例えば、Ａ１点の値を代用して、Ａ２点、Ｂ１点、Ｂ２点の値を求める。図11は分点（２点の場合は両側の平均値）を計算する方法である。例えば、Ａ１点とＣ１点との平均値としてＢ１点を、また、Ａ３点とＣ３点との平均値としてＢ３点を求める。そして、Ａ１点とＡ３点との平均値としてＡ２点を、また、Ｂ１点とＢ３点との平均値としてＢ２点を求める。

2

【０００４】しかし、図10のように同じデータを繰り返して補間するのでは、画素の構造が見えてくるだけであり、また、図11のように単に平均値をとっても、医用画像のように階調を有する画像には画質的には受入れられるレベルにはならない。そこで、医用画像には、注目点の近傍の（$m_x \times m_y$）の領域（これをマスクサイズという）の原画像データを用いて、より複雑な補間演算を行う方法が知られている。

【０００５】以下にかかる補間演算の従来例１，２について述べる。

従来例１：今、（$X_0 \times Y_0$）＝（８×８）の画像データを、マスクサイズ（$m_x \times m_y$）＝（４×４）で、２倍（補間倍率Ｎ＝２）に補間する場合を例にとる。原画像データと、補間で求める画像データとの位置関係は、図12のようになる。図中●が原画像データ、△が補間で求めるデータである。そして、補間で求めるデータのうち、◎を注目点とすると、図中の枠内がマスクサイズとなる。

【０００６】この場合、

Ｐ（ｉ，ｊ）：原画像データ　（$1 \leqq i \leqq 8$，$1 \leqq j \leqq 8$）

Ｒ（ｋ，ｌ）：補間後のデータ（$1 \leqq k \leqq 16$，$1 \leqq l \leqq 16$）

となる。これを実現するハードウェアブロック図は図13のようになり、画像データ入力部101 、入力フレームメモリ（ＦＭ１）102 、補間演算処理部103 、出力フレームメモリ（ＦＭ２）104 、画像データ出力部105 よりなる。尚、フレームメモリとは全画面のデータを格納し得るメモリを意味する。

【０００７】先ず画像データ入力部101 より図12の全●点の原画像データＰ（ｉ，ｊ）を入力フレームメモリ102 に格納する。次に図12の各△点について、その演算に必要な近傍の４×４の●点（16点）のデータを入力フレームメモリ102 より読出して、補間演算処理部103 にて所定の関数を用いて演算し、結果を出力フレームメモリ104 に書込む。

【０００８】すなわち、図12の◎を注目点とすると、◎点の演算に必要なデータ16点（枠内の●点）を読出して、演算処理する。この後、出力フレームメモリ104 から補間後のデータＲ（ｋ，ｌ）を読出して画像データ出力部105 より出力する。従って、全原画像データを格納する入力フレームメモリ、補間後のデータを格納する出力フレームメモリが必要となる上に、処理も１点（図12の◎点）を求めるのに16点（図12の枠内の●点）ものデータを入力フレームメモリから読出さなければならない。原画像が８ビットの階調性を有しているとなれば、８ビット×16点＝ 128ビット入力で、８ビット出力の得られる高速なデータ処理回路が必要となる。また、補間演算処理部の構成が極めて複雑になり、処理時間も増大

する。

【0009】従来例2：別の従来例として、従来例1と同様に（X_0 ×Y_0 ）＝（8×8）の画像データをマスクサイズ（m_x ×m_y ）＝（4×4）で補間する場合を例にとるが、1次元の2倍補間を第1及び第2の方向（縦横）に2回に分けて行う例を考える。原画像データと、補間で求める画像データとの位置関係は、図14のようになる。

＊【0010】図中の記号の意味は次の通りである。

●：原画像データ

△：1回目の補間で求めるデータ

◎：補間で求めようとしている注目点

◆：△のうち、◎を求めるために必要な中間データ

▽：2回目の補間で求めるデータ

この場合、

＊

P（i，j）：原画像データ　　（1≦i≦8，1≦j≦8）

V（i，l）：1回目の補間後の中間データ（1≦i≦8，1≦l≦16）

R（k，l）：2回目の補間後の最終データ（1≦k≦16，1≦l≦16）

となる。

【0011】また、補間演算のための関数としては、下記のE〔a，b，c，d〕、O〔a，b，c〕を用いる。

E〔a，b，c，d〕：　一列に並んだ点a，b，c，dのデータを用いて点bと点cとの間の点の補間値を与える（偶数補間）

O〔a，b，c〕　　：　一列に並んだ点a，b，cのデータを用いてb点自身の補間値を与える（奇数補間）

これを実現するハードウェアブロック図は図15のようになり、画像データ入力部201 、入力フレームメモリ（FM1）202 、補間演算処理部203 、中間処理フレームメモリ（FM3）204 、出力フレームメモリ（FM2）205 、画像データ出力部206 よりなる。

【0012】入力フレームメモリ202 は原画像データの格納用、中間処理フレームメモリ204 は1回目の補間後の中間データの格納用、出力フレームメモリ205 は2回目の補間後の最終データの格納用となる。補間演算処理部203 は、4点あるいは3点の入力に対し、偶数補間／奇数補間（関数E／Oの演算）を行う機能を果たす。

【0013】先ず画像データ入力部201 より原画像データP（i，j）を入力フレームメモリ202 に格納する。次に第1の方向（縦方向）に連続した4点のデータを入力フレームメモリ202から読出して、補間演算処理部203 にて関数E／Oの演算を施し、結果を中間処理フレームメモリ204 に書込む。そして、これを繰り返して、中間処理フレームメモリ204 に中間データV（i，l）を得る。

【0014】次に中間処理フレームメモリ204 から第2の方向（横方向）に連続した4点のデータを読出して、補間演算処理部203 にて関数E／Oの演算を施し、結果を出力フレームメモリ205 に書込む。そして、これを繰り返して、出力フレームメモリ205 に最終データR（k，l）を得る。この後、出力フレームメモリ205 から補間後のデータR（k，l）を読出して画像データ出力部206 より出力する。

【0015】しかし、この例においては、補間演算処理部は4点ないし3点のデータより演算を行う機能を持てばよいが、中間処理フレームメモリも含め、3つのフレームメモリが構成上必要となる。原画像が8ビットの階調性を有しているとなれば、画像バッファとして8×16×8ビット（原画像データの2倍）の中間処理フレームメモリを必要とする。ここでは原画像データが8×8×8ビットと比較的小さい例を示したが、実際の画像データ、特に医用の分野においては、2000×2000×12ビット等と、大きな情報量を有する画像を扱うので、このようなフレームメモリを必要とすることは、回路の規模や複雑さを膨らませる要因となる。

【0016】また、補間演算処理部の共通化はできるものの、処理を時間的に2度に分けているため、時間がかかるばかりか、補間演算処理部に必要なデータを随時供給しなければならないので、フレームメモリにもランダムアクセス性を持たせなければならず、回路構成を大きくしてしまう。

【0017】

【発明が解決しようとする課題】本発明は、上記の従来例の問題点に鑑み、必要となる画像メモリを少なくし、かつ、同様の処理を高速で実現させることができるようにすることを目的とする。

【0018】

【課題を解決するための手段】このため、本発明は、2次元画像データを1次元の補間を第1及び第2の方向に2度行うことによって補間するに際し、入力された画像データを第1の方向に補間演算処理し、この第1の方向の補間演算処理が第2の方向の補間演算処理が可能となるところまで進んだ時点から並行して第2の方向に補間演算処理することを特徴とする画像データ補間方法を提供する。

【0019】また、画像データ補間装置としては、画像データ入力部と、入力された画像データを第1の方向に補間演算処理する第1の補間演算処理部と、第1の方向の補間演算処理が第2の方向の補間演算処理が可能となるところまで進んだ時点から並行して第2の方向に補間演算処理する第2の補間演算処理部と、第2の方向の補間演算処理を終えた画像データを出力する画像データ出力部と、を設けて構成される。

【0020】このとき、第2の方向を画像データ出力方向と一致させるか、第1の方向を画像データ出力方向と

5

一致させるかする。

【0021】

【作用】すなわち、2回の補間を行うにあたって、第1の方向の補間を行いながら、第2の方向の補間を行うのである。このとき、2回目の補間の方向（第2の方向）を画像データ出力方向と一致させた場合と、1回目の補間の方向（第1の方向）を画像データ出力方向と一致させた場合とで、次のように相違する。

【0022】第2の方向が画像データ出力方向と一致している場合、（第2の方向の原画像データ数：X_0）×（第1の方向の補間に必要なデータ数：m_Y）×（原画像データの階調の深さ）のバッファメモリの容量で、2次元の補間を実現できる。上記の従来例2で言えば、8×4×8ビットのバッファメモリで補間を行うことができる。

原画像データ	横	X_0	： 1024（第2の方向）
	縦	Y_0	： 1024（第1の方向）
マスクサイズ	横	m_X	： 4
	縦	m_Y	： 4
補間後のデータ	横	X_C	： $2048=2\times X_0$
	縦	Y_C	： $2048=2\times Y_0$
原画像の値		$P(i,j)$	： $1\leqq i\leqq X_0$, $1\leqq j\leqq Y_0$
中間画像の値		$V(i,l)$	： $1\leqq i\leqq X_0$, $1\leqq l\leqq Y_C$
補間画像の値		$R(k,l)$	： $1\leqq k\leqq X_C$, $1\leqq l\leqq Y_C$
バッファメモリの値		$B(s,t)$	： $1\leqq s\leqq X_0$, $1\leqq t\leqq m_Y$

注1）P（i，j）の添字が最小値未満の時には、その添字が最小値をとったときと同じ値をとることとする。V、Rも同様。例として、P（－1，4）＝P（1，4）とする。

【0026】注2）　P（i，j）の添字が最大値を超える時には、その添字が最大値をとったときと同じ値をとることとする。V、Rも同様。例として、P（1026，4）＝P（1024，4）とする。これらの注は、周辺部近くのデータを補間するときに補間前のデータを仮想的に外側に補って求めることを表している。以降、補間演算の目的でこの外側に補った値をダミーデータという。

【0027】また、補間演算のための関数としては、下記のE〔a，b，c，d〕、O〔a，b，c〕を用いる。

E〔a，b，c，d〕：　一列に並んだ点a，b，c，dのデータを用いて点bと点cとの間の点の補間値を与える（偶数補間）

O〔a，b，c〕　：　一列に並んだ点a，b，cの※

$$y=E[a,b,c,d]=(1/16-1/8\beta)(a+d)+(7/16+1/8\beta)(b+c)$$

…ベル・スプライン法の偶数補間

$$x=O[a,b,c]=(1/4-1/4\beta)(a+c)+(1/2+1/2\beta)(b)$$

…ベル・スプライン法の奇数補間

尚、βは補間の性質を変化させるパラメータである。

6

＊【0023】第1の方向が画像データ出力方向と一致している場合、（第1の方向の補間されたデータ数：$X_0\times N$）×（第2の方向の補間に必要なデータ数：m_Y）×（原画像データの階調の深さ）のバッファメモリの容量で、2次元の補間を実現できる。上記の従来例2で言えば、8×2×4×8ビットのバッファメモリで補間を行うことができる。

【0024】

【実施例】以下に本発明の実施例を説明する。

実施例1：実施例1として、下記の補間を例に挙げ説明する（図2参照）。この例は、第1の方向を画像データ出力方向（横方向）とは異なる縦方向とし、第2の方向を画像データ出力方向と一致する横方向としたものである。

＊　【0025】

※データを用いて点b自身の位置の補間値を与える（奇数補間）

すなわち、下記のごとく、一列に並んだ点a，b，c，dの4点のデータを用いて点bと点cとの間の点の補間値yを得、また、点a，b，cの3点のデータを用いて点b自身の位置の補間値xを得るのである。

【0028】

尚、この補間演算では、原画像の点そのものの値も演算処理によって置換えられる。

【0029】関数E〔a，b，c，d〕及びO〔a，b，c〕の具体例としてベル・スプライン補間を挙げると、これらの関数は以下の式で表現される。

【0030】ここで用いているベル・スプライン補間は

1次元の補間で、以上は2倍に補間する場合であり、これを縦方向と横方向とに実施することで2次元の補間が実現される。これを実現するためのハードウェアブロック図は図1のようになる。すなわち、画像データ入力部301、入力フレームメモリ302、ラインバッファメモリ（BM1～BM4）303～306、第1の補間演算処理部307、リングバッファ308、第2の補間演算処理部309、画像データ出力部310よりなる。

【0031】ラインバッファメモリ（BM1～BM4）303～306により、4ライン分のデータを記憶することができる。尚、画像データ入力部よりの原画像データP（i，j）が格納される入力フレームメモリ302は、説明のために記述したが、画像データ入力部301からラインバッファメモリ（BM1～BM4）303～306へパイプライン的に順次データが送られるのであれば、入力フレームメモリ302はなくてもよい。

【0032】また、本実施例を実現するにあたってのフローチャートを図3に、1画像毎のタイミングチャートを図4に、1ライン毎のタイミングチャートを図5に示す。先ず入力フレームメモリから1ライン目のデータP（i，1）がP（1，1）、P（2，1）、P（3，1）・・・P（X_0，1）の順に読出され、BM1、BM2に格納される。これは図4のタイミングチャートの－3ラインサイクル目①に相当する。尚、図3のフローチャートにも図4及び図5のタイミングチャートに対応づけて見られるよう同じ丸付き番号をつけた。

【0033】続いて入力フレームメモリから2ライン目のデータP（i，2）が読出されると、BM3、BM2、BM1の値をBM4、BM3、BM2に移しながら、空いたBM1に2ライン目のデータを書込む。これは、図4のタイミングチャートの－1ラインサイクル目②に相当する。同様の手順で入力フレームメモリから3ライン目のデータP（1，3）がBM1に書込まれ始めると、図4のタイミングチャートの1ラインサイクル目③の動きとなる。

【0034】このラインの1点目のデータP（1，3）がBM1に書込まれた段階で、第1の方向（縦方向）の補間演算に必要なデータが揃うことになる。そこで、先ず、V（1，1）を求めるために関数O〔a，b，c〕の演算に必要な1点目のデータP（1，1）、P（1，1）、P（1，2）をBM4、BM3、BM2から順次読出して、関数O〔a，b，c〕の演算を行い、結果V（1，1）を得てリングバッファに書込む。これは図5のタイミングチャートの－5，－4画素サイクル目④に相当する。尚、この図5のタイミングチャートの中ではBM1をも読出すように書かれているが、これは4点を用いて補間する際（偶数補間）に必要となるので記したもので、3点で補間する際（奇数補間）には不要である。

【0035】2点目のデータP（2，3）が送られてくると、BM1に書込まれるが、前回の第1の方向の補間は第2の方向の補間のダミーデータ作成のためであるので、もう一度1点目のデータを読出して演算を行い、結果V（1，1）を得てリングバッファに書込む。リングバッファへの書込みはリングバッファ1に対して行われ、この書込みの際、リングバッファ1・2・3の元のデータはそれぞれ2・3・4に移動する。これは図5のタイミングチャートの－3，－2画素サイクル目に相当する。

【0036】続いて3点目のデータP（3，3）が送られてくると、BM1に書込まれるが、第1の方向の補間は2点目について行う。すなわち、2点目のデータP（2，1）、P（2，1）、P（2、2）を読出して関数O〔a，b，c〕の演算を行い、補間で求めた値V（2，1）をリングバッファに書込む。これは図5のタイミングチャートの－1，0画素サイクル目に相当する。

【0037】続いて4点目のデータP（4，3）が送られてくると、BM1に書込まれるが、第1の方向の補間は3点目について行う。すなわち、3点目のデータP（3，1）、P（3，1）、P（3、2）を読出して関数O〔a，b，c〕の演算を行い、補間で求めた値V（3，1）をリングバッファに書込む。これは図5のタイミングチャートの1，2画素サイクル目に相当する。

【0038】この段階で、ダミーを含めて4点分の中間データが得られ、第2の方向（横方向）の補間が可能になる。リングバッファには4点分の中間データV（1，1），V（1，1），V（2，2），V（3，1）が格納されているので、これらのデータを順次読出して、第2の方向の補間として、関数O〔a，b，c〕及び関数E〔a，b，c，d〕の演算を行い、R（1，1）、R（2，1）を得る。このようにして補間を済ませたデータは画像データ出力部に供給される。これが図4のタイミングチャートの1、2サイクル目⑤に相当する。

【0039】リングバッファにおいてはこの時点で一番古く書込まれたデータ（リングバッファ4の内容）が不要となり、次の第1の方向の補間後のデータが書込まれることになる。続いて5点目のデータP（5，3）が送られてくるが、以降は、同様の手順で処理がなされ、次々と補間を済ませたデータR（3，1）、R（4，1）・・・が画像データ出力部に供給されることになる。

【0040】R（2048，1）＝1ライン目の処理が終了すると、2ライン目の処理に移るが、これらのデータは1ライン目を求めるときに用いたデータ、すなわち既にBM1～4に格納されたデータを用いて得られるので、入力フレームメモリからの読出し・BMへの書込みはなく、読出したデータをこれまでと同じ手順で進めることで求めることができる。1ライン目と異なるのは第1の方向の補間に関数E〔a，b，c，d〕を用いる点である（図4のタイミングチャート1の⑥）。

【００４１】このように第２の補間演算処理部の演算方向（第２の方向）と画像データの出力方向とを合わせて、第１の方向の補間を行いながら第２の方向の補間を行うように構成したことで、バッファメモリの容量縮小が実現できた。また、画像データ入力部への原画像データの供給も通常の入力順序でなんら問題を生じないの＊で、ランダムアクセス性を持たせなくても済む。

【００４２】実施例２：実施例２として、下記の補間を例に挙げ説明する。この例は、第１の方向を画像データ出力方向（横方向）とは一致する横方向とし、第２の方向を画像データ出力方向と異なる縦方向としたものである。

原画像データ	横　X_0	：　1024（第１の方向）
	縦　Y_0	：　1024（第２の方向）
マスクサイズ	横　m_x	：　4
	縦　m_y	：　4
補間されたデータ	横　X_c	：　$2048=2\times X_0$
	縦　Y_c	：　$2048=2\times Y_0$
原画像の値	$P(i, j)$	：　$1\leqq i\leqq X_0$ ，$1\leqq j\leqq Y_0$
中間画像の値	$H(i, l)$	：　$1\leqq i\leqq X_c$ ，$1\leqq l\leqq Y_0$
補間画像の値	$R(k, l)$	：　$1\leqq k\leqq X_c$ ，$1\leqq l\leqq Y_c$
バッファメモリの値	$B(s, t)$	：　$1\leqq s\leqq X_c$ ，$1\leqq t\leqq m_y$

また、補間演算のための関数としては、前述のE〔a，b，c，d〕、O〔a，b，c〕を用いる。

【００４３】これを実現するためのハードウェアブロック図は図６のようになる。すなわち、画像データ入力部401 、入力フレームメモリ402 、リングバッファ403 、第１の補間演算処理部404 、ラインバッファメモリ（ＢＭ１～ＢＭ４）405 ～408 、第２の補間演算処理部409 、画像データ出力部410 よりなる。ラインバッファメモリ（ＢＭ１～ＢＭ４）405 ～408 は４ライン分のデータを記憶することができる。また、入力フレームメモリ402 は説明のために記述したが、画像データ入力部401 からラインバッファメモリ（ＢＭ１～ＢＭ４）405 ～408 へパイプライン的に順次データが送られるのであれば、入力フレームメモリ402 はなくてもよい。

【００４４】また、本実施例を実現するにあたってのフローチャートを図７に、１画像毎のタイミングチャートを図８に、１ライン毎のタイミングチャートを図９に示す。先ず入力フレームメモリから１ライン目のデータＰ（ｉ，１）が、Ｐ（１，１）、Ｐ（２，１）、Ｐ（３，１）・・・Ｐ（X_0 ，１）の順に読出され、リングバッファに格納される。これは図９のタイミングチャートの①に相当する。

【００４５】１画素サイクル目になると、第１の方向（横方向）の補間ができるだけのデータが揃うので、この時点で奇数補間（関数Ｏの演算）と偶数補間（関数Ｅの演算）とを実施しラインバッファメモリに格納する（１・２画素サイクル目）。次のデータがリングバッファに書込まれると、３・４画素サイクル目の処理が可能となり、順次進めて行くことによって、１ライン目の原画像データを第１の方向に補間処理の後、ラインバッファメモリへの格納が完了する。

【００４６】図８のタイミングチャートの１ラインサイクル目になると、第２の方向（縦方向）の補間が可能となる（図８のタイミングチャートの④、図７のフローチャートの④）。従って、ラインバッファメモリ１～４の内容を順次読出し、奇数補間を１ライン分実施して画像データ出力部に送り出し、その後偶数補間を１ライン分実施して画像データ出力部に送り出す（図８のタイミングチャートの⑤）。

【００４７】実施例１に対し、補間の順序を逆にした実施例２を示したが、このときには一度補間したデータをバッファメモリに蓄えているため、実施例１よりもバッファメモリ容量がＮ倍（本実施例では２倍）になっている。従って、メモリ容量という観点からみれば、画像データの出力方向の補間演算を２回目に行う方（実施例１）が優れている。

【００４８】これらの実施例１及び実施例２の各処理における１画像毎・１ライン毎のタイミングチャートを既に示した。ここで、１画像毎のタイミングチャート（図４と図８）を両者比較してみると、１画像の中での各ラインの処理手順については両者とも大きな違いがみられないが、１ライン毎のタイミングチャート（図５と図９）で処理内容を比較してみると、ラインの最初の処理に違いがみられる。

【００４９】実施例１は前処理としてデータを送り出す前にバッファメモリの書込み・読出しを行い、第１の方向の補間も３点分実行しておかなければならないが、実施例２の場合には前処理として３点分のデータのみを先読みしておけば、あとは送り出すタイミングに合わせて処理を進めることができる。従って、実施例２では、補間演算のタイミングが処理毎に異ならないので、タイミング生成部分の回路が実施例１よりも小規模で実現できる。

【００５０】もし、データの送り出しがそのデータを受取る側のタイミングで供給しなければならないときには、実施例２の方法はそのまま流用できるが、実施例１の方法ではデータの出力側にタイミング調整用のラインバッファを設けなければならない。補間された後のデー

タをバッファすることは、必要となるメモリ容量を増やすだけで、この点に関しては実施例２の方法が優れている。

【００５１】尚、以上では、２次元の補間について示したが、これより高次元の処理においても同様の処理順序を用いる（すなわち、データ出力方向と最終段の処理方向とを一致させる）ことによって、バッファメモリ容量の縮小が実現できる。また、画像の中からある特徴を持った部分を探し出すにも効果を生じる。画像欠陥を探す場合にマスク演算を施した後の情報を元にして、該当する部分か否かを判断する方法であれば、データを受取りながら、その前処理を行い、その時点で判断を加えていくことで、前処理を全て実施をする前に該当する部分を見つけることができる。

【００５２】

【発明の効果】以上説明したように本発明によれば、必要となる画像メモリを少なくし、かつ高速の処理を実現できるという効果が得られる。また、第２の補間の方向を画像データ出力方向と一致させることで、よりバッファメモリの容量縮小を実現できる。

【００５３】また、第１の補間の方向を画像データ出力方向と一致させることで、補間演算のタイミングが処理毎に異ならないので、タイミング生成部分の回路がより小規模で実現できる。

【図面の簡単な説明】

【図１】　本発明の実施例１を示すハードウェアブロック図

【図２】　同上実施例における原画像データを示す図

【図３】　同上実施例のフローチャート

【図４】　同上実施例の１画像毎のタイミングチャート

【図５】　同上実施例の１ライン毎のタイミングチャート

【図６】　本発明の実施例２を示すハードウェアブロック図

【図７】　同上実施例のフローチャート

【図８】　同上実施例の１画像毎のタイミングチャート

【図９】　同上実施例の１ライン毎のタイミングチャート

【図10】　一般的な補間方法１を示す図

【図11】　一般的な補間方法２を示す図

【図12】　従来例１として原画像データと補間で求める画像データとの位置関係を示す図

【図13】　従来例１のハードウェアブロック図

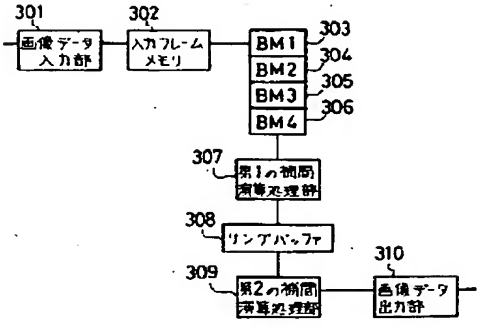
【図14】　従来例２として原画像データと補間で求める画像データとの位置関係を示す図

【図15】　従来例１のハードウェアブロック図

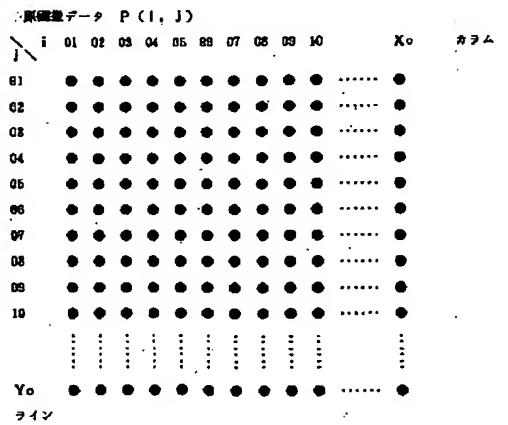
【符号の説明】

301　画像データ入力部
302　入力フレームメモリ
303 ～306　ラインバッファメモリ
307　第１の補間演算処理部
308　リングバッファ
309　第２の補間演算処理部
310　画像データ出力部
401　画像データ入力部
402　入力フレームメモリ
403　リングバッファ
404　第１の補間演算処理部
405 ～408　ラインバッファメモリ
409　第２の補間演算処理部
410　画像データ出力部

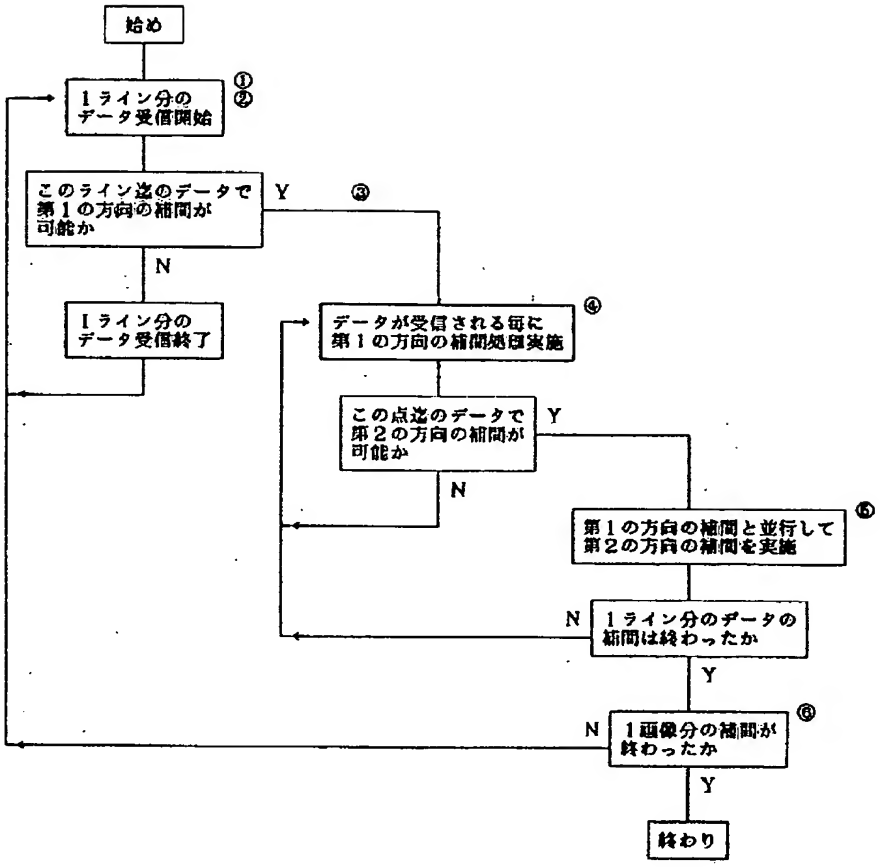
【図１】



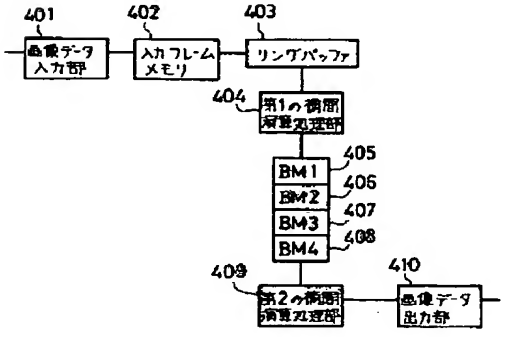
【図２】



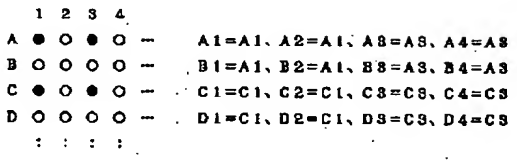
【図3】



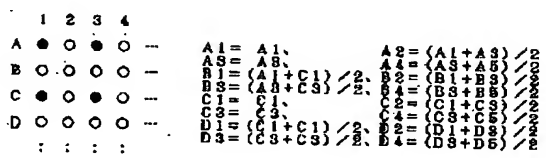
【図6】



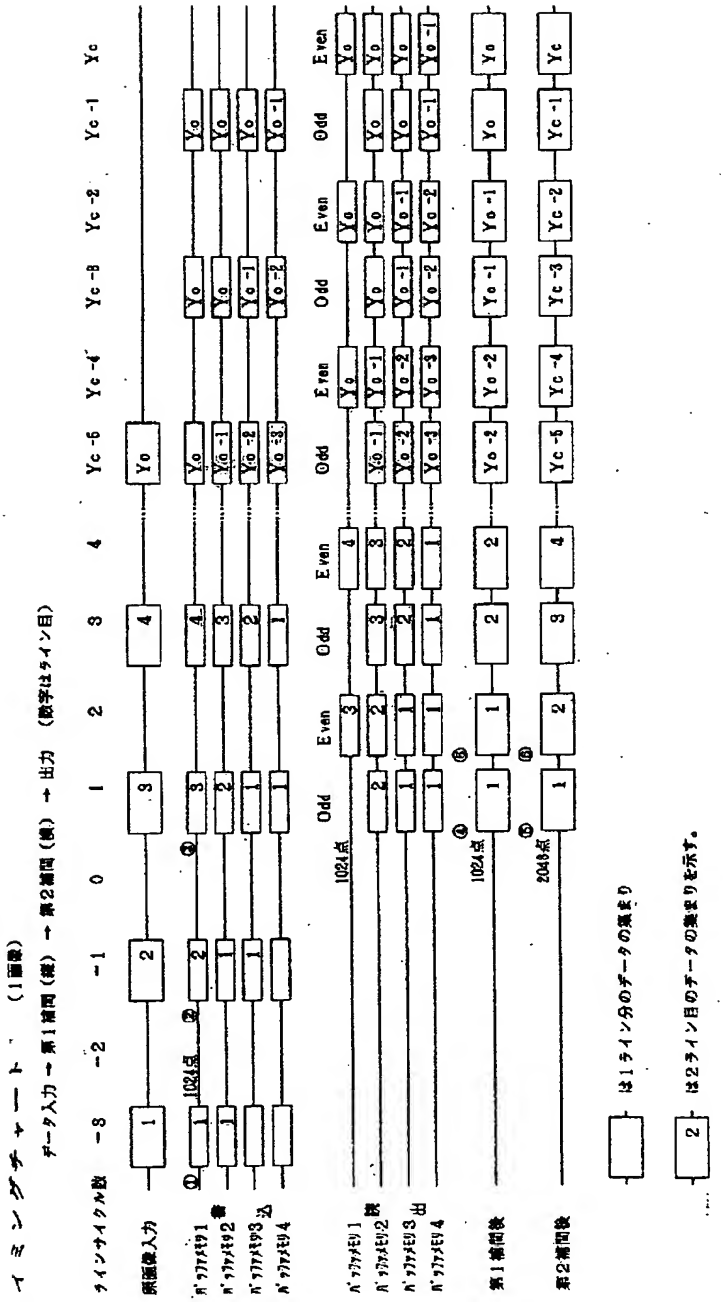
【図10】



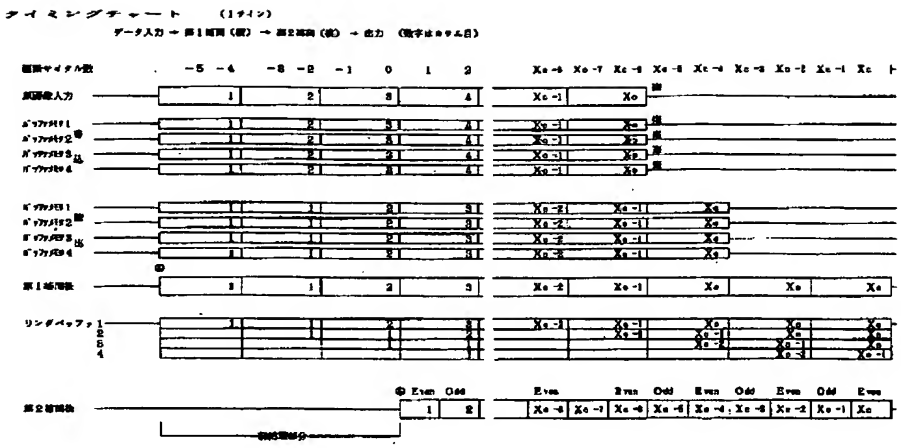
【図11】



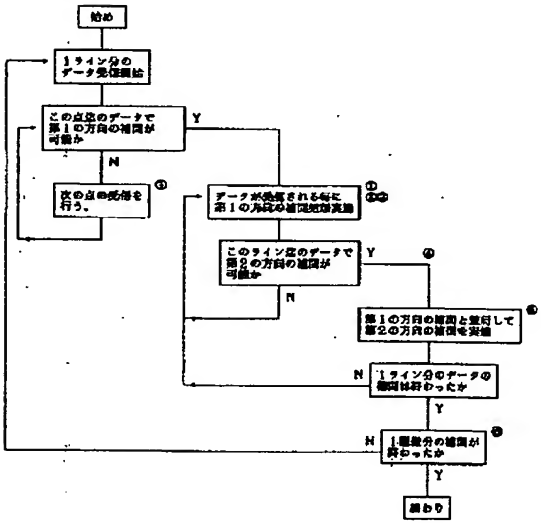
【図4】



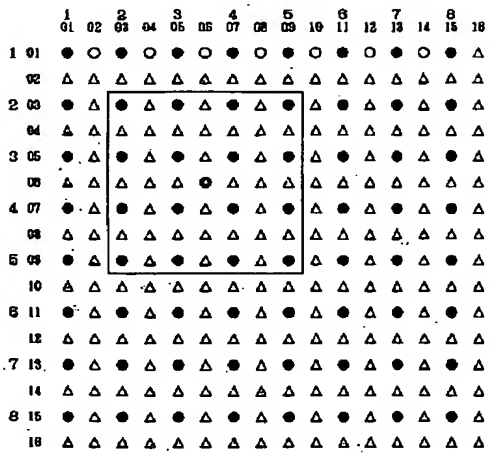
【図５】



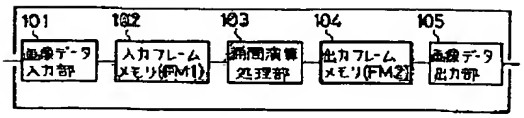
【図７】



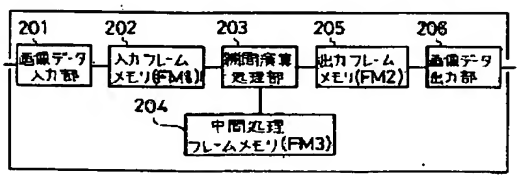
【図１２】



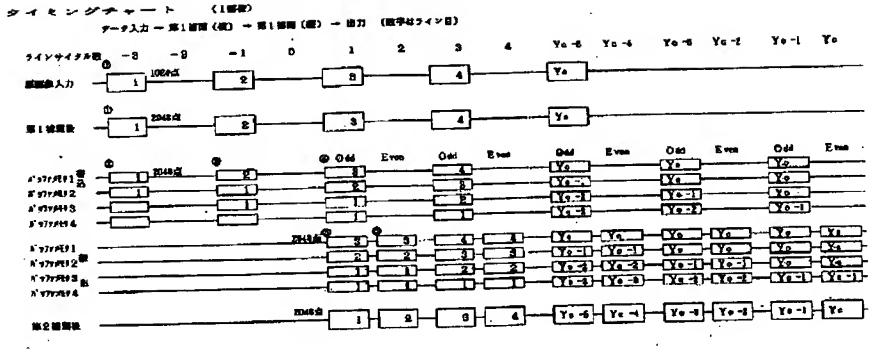
【図１３】



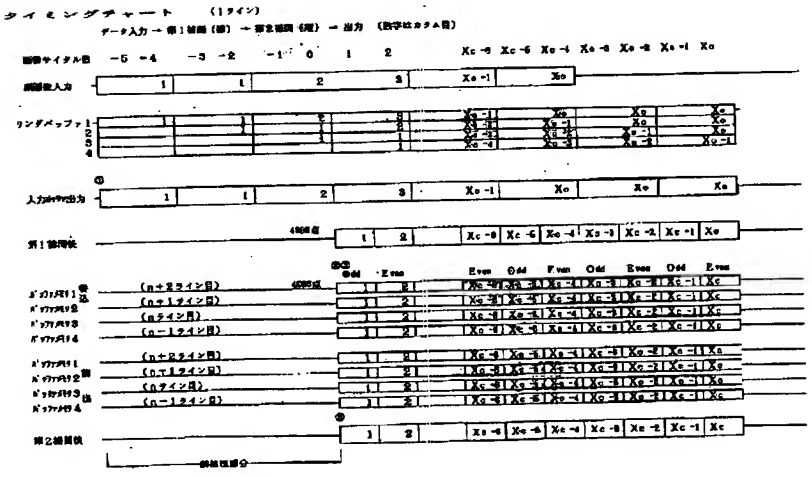
【図１５】



【図８】



【図９】



【図14】